CLAIMS:

What is claimed is:

1. A method of queuing threads among a plurality of processors in a multiple processor system having a plurality of multi-processor modules, the method comprising the computer implemented steps of:

receiving a first thread to be processed;

identifying the first thread as part of an existing process; and

performing a search for an idle processor, wherein the search is restricted to processors of a first multi-processor module associated with the existing process.

2. The method of claim 1, further comprising:

assigning the first thread to a queue dedicated to the first multi-processor module.

3. The method of claim 2, further comprising:

identifying the first multi-processor module as associated with the existing process.

4. The method of claim 3, wherein the step of identifying the first multi-processor module further comprises:

maintaining a record of processes having threads executed by a processor of the first multi-processor module during a predetermined preceding interval.

5. The method of claim 1, further comprising:

identifying one of the processors as an idle processor; and

assigning the first thread to a local run queue associated with the idle processor.

6. The method of claim 1, wherein the step of identifying further comprises:

reading attribute information of the first thread.

7. A method of load balancing in a multiple processor system having a plurality of multi-processor modules, the method comprising the computer implemented steps of:

performing, by an idle processor of a first multi-processor module, a first attempt at a thread steal from a local run queue of a processor located on the first multi-processor module for reassignment of a thread to a local run queue of the idle processor; and

responsive to failure of the first attempt, performing a second attempt at a thread steal from a dedicated queue associated with a second multi-processor module.

8. The method of claim 7, further comprising:

evaluating a criterion associated with the second multi-processor module; and

responsive to evaluating the criterion, determining if a thread is to be reassigned from the dedicated queue to the local run queue of the idle processor.

9. The method of claim 7, further comprising:

reassigning a thread of the dedicated queue to the local run queue of the idle processor.

10. The method of claim 7, further comprising:

responsive to failure of the second attempt, performing a third attempt at a thread steal from a local run queue associated with a processor of the second multi-processor module for reassignment of a thread to the local run queue of the idle processor.

11. A method of load balancing processors in a multiple processor system having a plurality of multi-processor modules, the method comprising the computer implemented steps of:

comparing a thread load of a first queue dedicated to a first multi-processor module with a thread load of a second queue dedicated to a second multi-processor module; and

reassigning a thread of the first queue to the second queue.

12. The method of claim 11, wherein the step of comparing further comprises:

determining a difference between the thread load of the first queue and the thread load of the second queue, reassigning the thread responsive to evaluating the difference as greater than a threshold.

13. A computer program product in a computer readable medium for queuing threads in a multiple processor system having a plurality of multi-processor modules, the computer program product comprising:

first instructions for receiving a first thread to be processed; and

second instructions for assigning the first thread to a first queue dedicated to a first multi-processor module of a plurality of multi-processor modules.

14. The computer program product of claim 13, further comprising:

third instructions for identifying a process associated with the first thread, wherein the second instructions identify threads of the process assigned to the first multi-processor module.

15. The computer program product of claim 13, further comprising:

third instructions for comparing a thread load of the first queue with a thread load of a second queue dedicated to a second multi-processor module of the plurality of multi-processor modules; and

fourth instructions for reassigning the first thread to the second queue.

16. The computer program product of claim 13, further comprising:

third instructions for reassigning the first thread to a second queue dedicated to a processor of a second multi-processor module of the plurality of multi-processor modules.

17. A multiple processor data processing system for executing multi-threaded processes, comprising:

a memory that contains a scheduler as a set of instructions;

a first multi-processor module; and

a second multi-processor module, wherein the scheduler, responsive to execution of the set of instructions, is adapted to receive a thread and assign the thread to a queue dedicated to the first multi-processor module.

18. The data processing system of claim 17, wherein the first multi-processor module comprises a plurality of central processing units disposed on a first chip, and the second multi-processor module comprises a plurality of central processing units disposed on a second chip.

19. The data processing system of claim 17, wherein the first multi-processor module is a simultaneous multi-threading central processing unit, and the second multi-processor module is a simultaneous multi-threading central processing unit.

20. The data processing system of claim 17, wherein the scheduler identifies a second thread of a process associated with the first thread, and the second thread is assigned to the queue.